On the fundamental limitations of performance for distributed decision-making in robotic networks

Federico Rossi Marco Pavone

***Abstract*— This paper studies fundamental limitations of performance for distributed decision-making in robotic networks. The class of decision-making problems we consider encompasses a number of prototypical problems such as average-based consensus as well as distributed optimization, leader election, majority voting on a limited number of options, MAX, MIN, and logical formulas. We first propose a formal model for distributed computation on robotic networks that is based on the concept of I/O automata and is inspired by the Computer Science literature on distributed computing clusters. Then, we present a number of bounds on time, message, and byte complexity, which we use to discuss the relative performance of a number of approaches for distributed decision-making. From a methodological standpoint, our work sheds light on the relation between the tools developed by the Computer Science and Controls communities on the topic of distributed algorithms.**

I. Introduction

Decentralized decision-making in robotic networks is a ubiquitous problem, with applications as diverse as state estimation [1], formation control [2], and cooperative task allocation [3]. In particular, the consensus problem, where the nodes in a robotic network have to agree on some common value, has received significant attention in the last decade following the works in [4], [5]. Most recent efforts in the Controls community have primarily focused on studying the properties and fundamental limitations of *average-based* consensus, a subclass of the consensus problem where nodes average their status with their neighbors at each (continuous or discrete) time step [6]. In these works, the dominant performance metric is time complexity, i.e., convergence time. In contrast, the computer science community has mainly focused its attention on the complementary notion of communication complexity, and "communication-optimal" algorithms for selected consensus problems are now known [7]. Indeed, time and message complexity are not the only relevant performance metrics for robotic networks: for instance, *bandwidth utilization* can be modeled via the overall number of *bytes* exchanged by an algorithm.

Despite the large interest in consensus problems in the last decade, little attention has been devoted to the problem of studying fundamental limitations of performance with respect to time and communication (in its broadest sense) complexity. In [6] the author proposes a lower bound on the time complexity of *average-based* consensus algorithms; several lower bounds on the *message* complexity of specific instances of the consensus problem are known in the CS literature [7], but no comprehensive study for more general distributed decision-making problems is available.

This motivates our work: in this paper, we explore tight lower bounds on the complexity of a large class of consensus problems with respect to metrics relevant to robotic systems. We mainly restrict our analysis to static networks, but also provide some extensions to time-varying network topologies. The contribution of our work is threefold: we extend results from the computer science literature to a broad class of distributed decision-making problems (collectively referred to as generalized consensus) relevant to the control systems and robotic community; we also present a unified complexity theory for generalized consensus on static networks, identifying lower bounds on performance for metrics relevant to robotic systems. Finally, we discuss algorithms that simultaneously make one or more of these bounds tight.

The paper is structured as follows. In Section II we propose a formal model for robotic networks, complexity measures and a rigorous definition of the consensus problem that encompasses problems including mean and weighed mean as well as voting. In Section III we present lower bounds on the time, message, and byte complexity of the consensus problems both for *sparse* and for *dense* networks. In Section IV we show tightness of these lower bounds under mild assumptions. Finally, in Section V we draw our conclusions and we discuss directions for future work.

II. Problem Setup

In this section we discuss the network model and we define the distributed consensus problem we will study in this paper. A simplified version of our model has first been introduced by the authors in [8].

A. Agent model

An agent in a robotic network is modeled as an input/output (I/O) automaton, i.e., a labeled state transition system able to send messages, react to received messages and perform arbitrary internal transitions based on the current state and on any messages received. A precise definition of I/O automaton is provided in [7, pages 200-204] and is omitted here in the interest of brevity. All nodes are identical except for a unique identifier (UID - for example, an integer). The time evolution of each node in the graph G is characterized by two key assumptions:

- **Fairness assumption**: the order in which transitions happen and messages are delivered is not fixed a priori. However, any enabled transition will *eventually* happen and any sent message will *eventually* be delivered.
- **Non-blocking assumption**: every transition is activated within l time units of being enabled and every message is delivered within d time units of being dispatched.

Essentially, the fairness assumption states that every node will have an opportunity to perform transitions, while the non-blocking assumption gives timing guarantees (but no synchronization). We refer the interested reader to [7, pages 212-215] for a detailed discussion of these assumptions. We argue here that these are *minimal* assumptions for most real-world robotic networks.

Federico Rossi and Marco Pavone are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305, {frossi2, pavone}@stanford.edu

B. Network model

A *robotic network* with n agents is modeled as a *connected*, *undirected* graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the node set, and $E \subset V \times V$, the edge set, is a set of *unordered* node pairs modeling the availability of a communication channel. We will denote with $|G|$ the number of nodes in the graph G. Two nodes i and j are neighbors if $(i, j) \in E$. The neighborhood set of node $i \in V$, denoted by N_i, is the set of nodes $j \in V$ neighbors of node i. Our model is *asynchronous*, i.e., computation steps within each node and communication are, in general, asynchronous.

This paper focuses on *static networks*, i.e., robotic networks where the edge set does not change during the execution of the algorithm. Given a node set with n nodes, we will consider two classes of graphs:

- **Sparse graphs**: that is graphs where the number of edges $|E|$ is less than $n \log n$.
- **Dense graphs**: that is graphs where the number of edges $|E|$ is larger than or equal to $n \log n$.

Note that for any connected graph with n nodes, $n - 1 \leq |E| \leq \binom{n}{2}$. From a practical standpoint, sparse and dense graphs manifest themselves in different cyber-physical problems and give rise to different issues. In dense graphs (present e.g. in short-range formation control and rendezvous problems) time complexity is typically not an issue; on the other hand, message and byte complexity have to be carefully kept under control to avoid excessive bandwidth utilization and minimize message collisions. Especially for large graphs, it is crucial to ensure that agents only communicate with a small subset of their neighbors, even if many are available. On the other hand, in sparse graphs (which typically manifest themselves in patrolling and deployment applications, where large inter-agent distances are desirable) message complexity is not an issue; efficient routing of information, on the other hand, is crucial to ensure good time performance.

C. Model of computation

At a general level, in this paper we focus on decision-making problems where each node i, $i \in \{1, \ldots, n\}$, in the robotic network is endowed with an initial value x_i and should output the value of a function of all initial values. In other words, each agent, after exchanging messages (with any content) with its neighbors and performing internal state transitions, should output $f(x_1, \ldots, x_n)$ for some computable function f, referred to as *consensus* function. In the reminder of this section we formalize the notions of consensus functions and of decentralized algorithms.

1) Consensus functions: In this paper we consider functions defined over *totally ordered sets*, that is we assume that the initial conditions x_i, $i \in \{1, \ldots, n\}$, belong to a set $\mathcal{X}$ equipped with a binary relation (denoted with $\leq$) which is transitive, antisymmetric, and total. Two sets of initial conditions $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ are said to be *order-equivalent* if $a_i < a_j \Leftrightarrow b_i < b_j$. The set of initial conditions $\mathcal{C}$ can be, for example, $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{R}^d$ (in the last case the total order could be the lexicographic order).

A consensus function is a *computable* [9] function $f : \mathcal{X}^n \mapsto \mathbb{R}$ that depends on *all* its arguments. More precisely, for each element $x = (x_1, \ldots, x_n) \in \mathcal{X}^n$ and for all $i \in \{1, \ldots, n\}$ one can find elements $x_i^{(1)} \in \mathcal{C}$ and $x_i^{(2)} \in \mathcal{C}$ such that

$$f(x_1, \ldots, x_i^{(1)}, \ldots, x_n) \neq f(x_1, \ldots, x_i^{(2)}, \ldots, x_n).$$

Loosely speaking, such choice of consensus function implies that each node is needed for the collective decision-making process.

For some of the results presented in this paper, we will need the following *refinements* of the notion of consensus function:

- **Locally-sensitive consensus function**: a consensus function that is sensitive to perturbations that preserve order. More precisely, let $\mathcal{I} = \{1, 2, \ldots, n\}$ be the set of node indices and let $\sigma : \mathcal{I} \mapsto \mathcal{I}$ be a permutation[1] (i.e., a bijective correspondence) over $\mathcal{I}$. Then, for each permutation σ over $\mathcal{I}$ there exists an element $x \in \mathcal{X}^n$ that is ordered with respect to σ, i.e., $x_{\sigma(i)} \leq x_{\sigma(j)}$ for all $i < j$ and such that for all $i \in \{1, \ldots, n\}$ there exist $x_{\sigma(i)}^{(1)}, x_{\sigma(i)}^{(2)} \in \mathcal{X}$ with the properties:
  1) $x_{\sigma(i)}^{(1)} \leq x_{\sigma(j)}$ and $x_{\sigma(i)}^{(2)} \leq x_{\sigma(j)}$ for all $i < j$,
  2) $x_{\sigma(j)} \leq x_{\sigma(i)}^{(1)}$ and $x_{\sigma(j)} \leq x_{\sigma(i)}^{(2)}$ for all $j < i$,
  3) $x_{\sigma(i)}^{(1)} = x_{\sigma(j)}$ if and only if $x_{\sigma(i)}^{(2)} = x_{\sigma(j)}$ for all $j \neq i$,
  4) $f(x_1, \ldots, x_{\sigma(i)}^{(1)}, \ldots x_n) \neq f(x_1, \ldots, x_{\sigma(i)}^{(2)}, \ldots x_n)$.

  (The first three properties ensure that $x_{\sigma(i)}^{(1)}$ and $x_{\sigma(i)}^{(2)}$ preserve the order of $x_{\sigma(i)}$, while the last property reflects local sensitivity.)
- **Extractive consensus function**: a consensus function such that for all $c = (x_1, \ldots, x_n) \in \mathcal{X}^n$ one has $f(x_1, \ldots, x_n) = x_j$ for some $j \in \{1, \ldots, n\}$.

The classes of locally-sensitive and extractive consensus functions are neither mutually exclusive nor collectively exhaustive (however, they represent, arguably, a very broad class of consensus functions of interest to applications). Loosely speaking, locally-sensitive consensus functions model problems where the decision-making process depends *continuously* on the initial values $\{x_i\}$, for example for average consensus

$$f(x) = c^T x,$$

where $\mathcal{X} = \mathbb{R}^n$, $x \in \mathcal{X}$, and c is a vector in $\mathbb{R}^n$, or for distributed optimization

$$f(x) = \operatorname*{argmin}_{z \in \mathbb{R}^n} \sum_{i=1}^{n} \varphi_i(z, x_i)$$

where the objective function is *parametrized* by x_i, under certain conditions on ϕ_i (consider for instance φ_i a SDP quadratic form parametrized by x_i). On the other hand, extractive consensus functions model leader-election problems or problems where it is desired to extract some statistics from the data, e.g., the median, MAX and MIN.

Finally, we introduce a *representation* property for consensus functions that will be instrumental to deriving fundamental limitations of performance in terms of amount of information exchanged. A locally-sensitive or extractive consensus function is hierarchically computable if it can be

[1] As an example, a particular permutation of the set $(1, 2, 3, 4)$ is $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$, and $\sigma(4) = 4$, which leads to the reordered set $(3, 1, 2, 4)$.

written as the composition of a commutative and associative binary operator $*$, that is

$$f(x_1, x_2, \ldots, x_n) = x_1 * x_2 * \ldots * x_n.$$

(The name is inspired by the observation that hierarchically computable functions can be computed with messages of small size on a *hierarchical* structure such as a tree.) All examples of consensus functions mentioned above are indeed hierarchically computable.

D. Model of communication

Nodes can communicate with their neighbors according to two communication schemes: *directional* and *local broadcast*. In the directional communication scheme a node sends messages to its neighbors sequentially. This is the case when nodes in the robotic network are equipped with efficient narrow-band, high-gain mechanically or electronically steerable antennas. In the local broadcast communication scheme, a node broadcasts a message to all its neighbors simultaneously. This is the typical case for nodes equipped with omnidirectional antennas.

E. Distributed algorithms

A distributed algorithm for a robotic network is, simply, a collection of *local* algorithms, one for each node of the network (of course, nodes can exchange messages). Nodes execute the same logical code. Each node is initialized with an initial condition $x_i \in \mathcal{X}$ and may or may not have knowledge of the consensus function (in other works, each node in the robotic network is aware of the task that the robotic network should accomplish). A distributed algorithm correctly computes a given consensus function if, given an input $x \in \mathcal{X}^n$, *each* node outputs the correct value of the consensus function $f(x_1, \ldots, x_n)$, and terminates [7] in *finite* time. Note that, for asynchronous executions, termination is not simultaneous.

A particular class of distributed algorithms that will be instrumental to derive some of the results is represented by comparison-based algorithms, defined next.

Definition II.1 (Comparison-based algorithms ([10], [11]))**.** *A distributed algorithm is comparison-based if each local algorithm manipulates the subset of initial values that are locally known only via the three Boolean operators $<$,$>$, and $=$ (recall that the set of initial values is a totally ordered set). Accordingly, all internal transitions (including message generation) only depend on the* order *of the initial values known to the local algorithm, as opposed to their numerical value.*

F. Distributed consensus in robotic networks

Given a consensus function f and a robotic network G where n nodes communicate either with a unidirectional or local broadcast scheme, the problem of distributed consensus is to find a distributed algorithm a that correctly computes $f(x)$ for all $x \in \mathcal{X}^n$. Given a consensus function f, let $\mathcal{A}$ be the set of all distributed algorithms that solve the distributed consensus problem and that terminates for all $x \in \mathcal{X}^n$ in at most R rounds (for synchronous executions) or after $R\,(l+d)$ time units (for asynchronous executions). We omit the explicit dependency of $\mathcal{A}$ on f and R for notational simplicity. Note that R can be an arbitrarily large number, hence this assumption is not limiting from a practical standpoint; however, it will be key to deriving some of our results.

G. Complexity measures

The following definitions naturally capture the notions of time and communication complexity and are widely used in the theory of distributed algorithms [7].

Let $\mathcal{G}$ be a set of graphs with node set $V = \{1, \ldots, n\}$ (we will be specifically interested in the class of sparse graphs and in the class of dense graphs). For a given graph $G \in \mathcal{G}$, let $\mathcal{F}(a, x, G)$ be the set of *fair executions* for an algorithm $a \in \mathcal{A}$ and set of initial conditions $x \in \mathcal{X}^{|G|}$ (a fair execution is an execution of an algorithm that satisfies the fairness and non-blocking assumptions stated above).

1) Time complexity: To measure execution time, we assume that a distributed algorithm starts at time $t = 0$. Time complexity is defined as the infimum worst-case (over initial values and fair executions) completion time of an algorithm. Rigorously, the time complexity for a given consensus function f with respect to the class of graphs $\mathcal{G}$ is

$$\mathrm{TC}(f, G) := \inf_{a \in \mathcal{A}} \sup_{G \in \mathcal{G}} \sup_{x \in \mathcal{X}^{|G|}} \sup_{\alpha \in \mathcal{F}(a,x,G)} T(a, x, \alpha, G),$$

where $T(a, x, \alpha, G)$ is the first time when all nodes have computed the correct value for the consensus function f and have stopped. The order of the inf-sup operands in the above expression is naturally induced by our definitions. By dropping the leading $\inf_{a \in \mathcal{A}}$, one recovers the time complexity of a given algorithm a for a given consensus function f. In our asynchronous setting, time complexity is expressed in multiples of $l + d$, defined in section II-B (see also [7]). We will henceforth refer to $(l + d)$ as a *time unit*. Note that $(l + d)$ is a (tight) upper bound on the actual time required to generate and deliver a message.

2) Message complexity for directional communication: Message complexity is similarly defined as the infimum worst-case (over initial values and fair executions) *number* of messages exchanged by an algorithm before completion. Rigorously, the message complexity for a given consensus function f with respect to the class of graphs $\mathcal{G}$ is

$$\mathrm{MC}(f, \mathcal{G}) = \inf_{a \in \mathcal{A}} \sup_{G \in \mathcal{G}} \sup_{x \in \mathcal{X}^{|G|}} \sup_{\alpha \in \mathcal{F}(a,x,G)} M(a, x, \alpha, G),$$

where $M(a, x, \alpha, G)$ is the number of messages exchanged between time $t = 0$ and time $t = T(a, x, \alpha, G)$.

It is important to note that the *type* of messages exchanged *depends* on the algorithm. In average-based consensus algorithms, nodes typically exchange their state, a real number. In algorithms such as the well-known Gallager, Humblet and Spira (GHS) algorithm [12], nodes exchange a wide range of logical commands establishing hierarchical relationships, informing neighbors about the progress of the algorithm, and requiring them to perform edge searches [13]. In flooding algorithms [7], a single message may contain information from up to $n - 1$ nodes. However, as far as message complexity is concerned, each message *counts the same*, regardless of its type and size.

3) Byte complexity for directional communication: In many instances, message size plays a critical role in the energy needed for information transmission. To capture this aspect, in this paper we define byte complexity as the infimum worst-case (over initial values and fair executions)

overall size (in bytes) of all messages exchanged by an algorithm before its completion. Rigorously, the byte complexity for a given consensus function f with respect to the class of graphs $\mathcal{G}$ is

$$\mathrm{BC}(f,\mathcal{G}) := \inf_{a\in\mathcal{A}} \sup_{G\in\mathcal{G}} \sup_{x\in\mathcal{X}^{|G|}} \sup_{\alpha\in\mathcal{F}(a,x,G)} B(a,x,\alpha,G),$$

where $B(a,x,\alpha,G)$ is the overall size (in bytes) of all messages exchanged between time $t=0$ and time $t=T(a,x,\alpha,G)$.

4) Message and byte complexity for local broadcast communication: The definitions of message and byte complexity for local broadcast communication parallels the definitions of message and byte complexity for directional communication. Rigorously, the broadcast message complexity for a given consensus function f with respect to a class of graphs $\mathcal{G}$ is

$$\mathrm{bMC}(f,\mathcal{G}) = \inf_{a\in\mathcal{A}} \sup_{G\in\mathcal{G}} \sup_{x\in\mathcal{X}^{|G|}} \sup_{\alpha\in\mathcal{F}(\alpha,x,G)} bM(a,x,\alpha,G)$$

where $bM(a,x,\alpha,G)$ is the overall number of *broadcast* messages exchanged between time $t=0$ and time $t=T(a,x,\alpha)$. Analogously, the broadcast byte complexity for a given consensus function f with respect to a class of graphs $\mathcal{G}$ is

$$\mathrm{bBC}(f,\mathcal{G}) = \inf_{a\in\mathcal{A}} \sup_{G\in\mathcal{G}} \sup_{k\in\mathcal{X}^{|G|}} \sup_{\alpha\in\mathcal{F}(\alpha,x,G)} bB(a,x,\alpha,G)$$

where $bB(a,x,\alpha,G)$ is the size (in bytes) of all *broadcast* messages exchanged between time $t=0$ and time $t=T(a,x,\alpha)$.

H. Discussion

It is of interest to compare our model of distributed consensus with the models for the consensus problem developed, respectively, by the Computer Science community and the Controls and Robotics community. In the Computer Science community, distributed consensus is typically defined as the task of computing via a distributed algorithm *any* function of a set of initial conditions such that the following three properties are fulfilled: *agreement* (no two processes decide on different values), *validity* (in absence of failures, if all agents start with the same value, then every agent decides on that value) and *termination* (all processes eventually decide) [7]. In the Controls and Robotics community, on the other hand, consensus is essentially a synonym for *average-based* consensus, where agents compute an asymptotic approximation of a weighted average of their initial conditions via local communication [6], [4], [14], [1], [2], [3]. Our model, thus, is more restrictive than the model considered in the Computer Science community (since the consensus function is required to fulfill some mild requirements), while it is (significantly) more general than the model considered in the Controls and Robotics community (since average-based consensus is a particular example of locally-sensitive consensus function defined over $\mathbb{R}^n$).

We mention that we also studied *storage complexity*, defined as the infimum worst-case (over initial values and fair executions) storage size required by every agent executing the consensus algorithm. We do not discuss this complexity notion here due to space limitation and because in most cases it is not a bounding factor. However, we report our results in the synoptic table (Table I) presented at the end of the paper.

In the remainder of the paper we discuss fundamental limitations of performance of the distributed consensus problem, in terms of fundamental scalings of the different complexity measures with respect to the network size (i.e., the number of nodes). We also discuss algorithms that, in many cases, recover such asymptotic bounds. Our asymptotic notation (e.g., $O(g(n))$ or $\Omega(g(n))$ is standard.

III. Lower Bounds on Achievable Performance for Distributed Consensus

In this section we present lower bounds for the complexity measures introduced in Section II-G. We will discuss the tightness of these bounds in Section IV.

A. Time complexity

A lower bound on time complexity can be obtained rather easily.

Proposition III.1 (Lower bound on time complexity)**.** *For a given consensus function f and class of graphs $\mathcal{G}$ with n nodes, TC$(f,\mathcal{G}) \in \Omega(n)$.*

Proof. Omitted due to space limitations. □

B. Message complexity

It this section we restrict our attention to either locally-sensitive or extractive consensus functions. Our strategy is to find first a lower bound for "dense" graphs (i.e., that becomes tight for dense graphs) and than a lower bound for "sparse" graphs (i.e., that becomes tight for sparse graphs). We start with the dense graph case.

Proposition III.2 (Lower bound on message complexity for dense graphs)**.** *For a given locally-sensitive or extractive consensus function f and class of graphs $\mathcal{G}$ with n nodes and $|E| \geq n\log n$, MC$(P,\mathcal{G}) \in \Omega(|E|)$.*

Proof. Computation of any consensus function f requires that *at least* one message is sent along *every* edge of a spanning subgraph of the network. If this was not true, there would exist two subsets of the nodes $V_1 \subset V$ and $V_2 = V\backslash V_1$ s.t. no messages are exchanged between nodes in V_1 and in V_2. Then, nodes in V_1 would have no information about the initial values of nodes in V_2 and vice versa. Since f depends on all initial values, this leads to a contradiction. Now, it can be shown that *any* computation problem that requires use of a *spanning subgraph* (i.e., at least one message sent along each of its edges) may require $|E|-1$ messages (and therefore $\Omega(|E|)$ messages) on a certain class of *almost complete* graphs [15]. This concludes the proof. □

We now turn our attention to a lower bound that becomes tight for sparse graphs. We first consider comparison-based algorithms, we will relax this assumption in Proposition III.4.

Lemma III.3 (Lower bound on message complexity for sparse graphs and comparison-based algorithms)**.** *Let f be a locally-sensitive or extractive consensus function and $\mathcal{G}$ be a set of graphs with n nodes and $|E| < n\log n$. Let $\mathcal{A}_c$ be the set of comparison-based algorithms that solve the distributed consensus problem and assume that one minimizes message complexity over the set $\mathcal{A}_c$ (we denote the result of such minimization as MC$(f,\mathcal{G})_{|\mathcal{A}=\mathcal{A}_c}$). Then MC$(f,\mathcal{G})_{|\mathcal{A}=\mathcal{A}_c} \in \Omega(n\log n)$.*

Proof. Consider the restriction to synchronous executions (since synchronous executions are a special case of asynchronous executions, a lower bound with respect to synchronous executions translate into a bound for the more general asynchronous case).

The proof is inspired by [10] and relies on the notion of c-symmetric rings. Consider a graph with a ring topology (i.e., the n nodes are lined up along a circle). A segment S on the ring is a sequence of consecutive nodes in the ring, in clockwise order. Two segments of the same length are said order-equivalent if the ordered vector of initial conditions of their respective nodes are order-equivalent. Let c be a positive constant. A ring is c-symmetric if, for every $l \in \mathbb{N}$ such that $\sqrt{n} \leq l \leq n$, and for every segment S of length l, there are at least $\lfloor cn/l \rfloor$ segments in the ring that are order-equivalent to S (including S itself).

We now study the message complexity of comparison-based algorithms on c-symmetric rings. To this purpose, without loss of generality [2], we consider comparison-based algorithms where at each time step (recall that we are considering *synchronous* executions) a node decides whether to send a message to its right neighbor, whether to send a message to its left neighbor, and whether to stop execution and decide on a consensus value. Every received message is stored in the receiver node's state. Every sent message contains the sender's entire state. Nodes can perform arbitrary internal transitions and have unlimited computational power. At each time step, the state of a node contains its initial value, its UID, and the history of messages exchanged with the neighbors. The initial conditions *include the nodes' UIDs* (totally ordered according to some binary relation $\leq$). Two states are order-equivalent if their initial conditions are order-equivalent.

It can be shown that (see [10]) (i) for any n, there exists a set of initial conditions such that there exists a c-symmetric ring for some $c > 0$, (ii) if a comparison-based algorithm exchanges $o(n \log n)$ messages on a c-symmetric ring, then every node receives information from nodes within distance k where $k < n/2$ (hence, from a *subset* of all nodes), and (iii) if a comparison-based algorithm exchanges $o(n \log n)$ messages on a c-symmetric ring, then every node i's state is order-equivalent to another node j's state at the end of the execution. More precisely, the k-neighborhoods of agents i and j (defined as the set containing the node and the $2k$ neighbors closest to it) contain agents with UIDs in identical order: such neighborhoods can not be distinguished by a comparison-based algorithm. This implies that the leader election problem has a message complexity $\Omega(n \log n)$ [10], since after $o(n \log n)$ message at least two agents are in states that are indistinguishable by a comparison-based algorithm.

We now apply these results to distributed consensus problems with locally-sensitive or extractive consensus functions. The lower bound $\Omega(n \log n)$ on message complexity directly applies to extractive consensus functions: any distributed consensus algorithm capable of extracting the initial value of a node with $o(n \log n)$ messages would also solve the leader election problem with $o(n \log n)$ messages, which would contrast with the aforementioned result for leader election.

Consider, now, locally-sensitive consensus functions. We proceed by contradiction, assuming that there exists a comparison-based distributed algorithm that solves the consensus problem with a locally-sensitive consensus function with message complexity $o(n \log n)$. Consider a given set of initial conditions, x, and let the nodes compute the value of the consensus function. Then, consider a set of initial conditions x' identical to x except that one of the node's initial values (and therefore the overall consensus value) is perturbed without changing the overall ordering of UIDs and node values (this is possible by the assumption of locally-sensitive consensus function). We next show that after $o(n \log n)$ messages at least one node would output the same consensus value it computed for initial condition x – a contradiction.

Let the nodes be arranged in a ring and let $x^{(1)}$ be an initial condition such that the ring is c-symmetric. By the contradiction hypothesis, after $o(n \log n)$ messages are exchanged, every node's state is order-equivalent to every other node's state and each node correctly outputs $f(x^{(1)})$ - call this execution $\alpha^{(1)}$. Note that, by fact (ii) above, each node has only received information from $2k + 1 < n$ neighbors, including itself. Now consider a pair u, v of nodes in order-equivalent states: there exists one node w that belongs to the k-neighborhood of u but does not belong to the k-neighborhood of v. Consider now an initial condition $x^{(2)}$ identical to $x^{(1)}$ except that the initial value of w is perturbed without changing the overall order of agents' values and UIDs, and such that $f(x^{(1)}) \neq f(x^{(2)})$ (this is always possible when the consensus function is locally-sensitive). Given initial condition $x^{(2)}$, under any execution $\alpha^{(2)}$ with $o(n \log n)$ messages, v's state will be identical (and not just order equivalent) to the state under execution $\alpha^{(1)}$, and therefore v will output $f(x^{(1)})$ as its consensus value (since by the contradiction hypothesis the algorithm terminates after $o(n \log n)$ messages). This is a contradiction. □

The bound in Lemma III.3 is instrumental to derive the desired lower bound over the much more general class of distributed consensus algorithms. Such bound requires that the set of initial values, i.e., $\mathcal{X}$ is "large enough". This requirement is automatically satisfied whenever $\mathcal{X}$ has infinite cardinality.

Proposition III.4 (Lower bound on message complexity for sparse graphs)**.** *Let f be a locally-sensitive or extractive consensus function and $\mathcal{G}$ be a set of graphs with n nodes and $|E| < n \log n$. Then there exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then MC$(f, \mathcal{G}) \in \Omega(n \log n)$.*

Proof. The key idea (conceptually identical to that in [10] and [11]) is to show that, perhaps surprisingly, if the cardinality of $\mathcal{X}$ is larger than a (very large) finite number, any distributed consensus algorithm in $\mathcal{A}$ executes on a small subset of $\mathcal{X}$ in a way that it is indistinguishable from the execution of a comparison-based algorithm. Lemma III.3 then applies and the claim follows.

As in the proof of Lemma III.3, without loss of generality, we consider synchronous executions and distributed consensus algorithms (referred to as *elementary* algorithm)where at

[2] Any consensus algorithm on a ring can be simulated by an algorithm in this class: in a generic algorithm, nodes perform transitions based on their states and the messages they receive, and the content of any message is the result of internal state transitions. We assume no bound on the nodes' computational power and no limitations on their internal transitions; therefore any lower bound on this class of algorithms is a lower bound on *all* consensus algorithms on rings.

each time step a node decides whether to send a message to its right neighbor, whether to send a message to its left neighbor, and whether to stop execution and decide on a consensus value. Every received message is stored in the receiver node's state. Every sent message contains the sender's entire state. Nodes can perform arbitrary internal transitions and have unlimited computational power. At each time step, the state of a node contains its initial value, its UID, and the history of messages exchanged with the neighbors. The initial conditions *include the nodes' UIDs* (totally ordered according to some binary relation $\leq$).

We introduce the definition of *indistinguishable* initial values. Consider two sets of initial conditions $x^{(1)}$ and $x^{(2)}$, whose elements are arranged in increasing order,that is $x_i^{(1)} \leq x_j^{(1)}$ and $x_i^{(2)} \leq x_j^{(2)}$ for $i \leq j$. Let σ be a permutation over the set of indexes $\mathcal{I} = \{1, \ldots, n\}$. We say that $x^{(1)}$ and $x^{(2)}$ are *indistinguishable* with respect to an algorithm $a \in \mathcal{A}$ if, for *any* permutation σ, the trace[3] of the execution with initial values $x^{(2)}$ (with indices permuted according to σ) can be obtained from the trace of the algorithm with initial values $x^{(1)}$ (with indices permuted according to σ) merely by substituting every occurrence of an element of $x^{(2)}$ with the element of corresponding order in $x^{(1)}$.

We claim that, if the set $\mathcal{X}$ of possible initial values is large enough, there exists a set $\mathcal{W} \subseteq \mathcal{X}$ with $|\mathcal{W}| \geq 2n-1$ such that *any* two n-subsets $\mathcal{U} \subset \mathcal{W}$ of size $|\mathcal{U}| = n$ are indistinguishable with respect to a given distributed consensus algorithm $a \in \mathcal{A}$. This claim follows from Ramsey's theorem [16, Theorem B]. Specifically, color every subset $\mathcal{U} \subset \mathcal{X}$ of size $|\mathcal{U}| = n$ so that indistinguishable sets share the same color. For any elementary algorithm, there are finitely many set of indistinguishable initial conditions. This is due to the facts that (i) there are finitely many permutations σ (specifically, $n!$), (ii) at each time step each node can make a finite number of decisions (specifically, 8), and (iii) the algorithm must terminate in a finite number of time steps (since the algorithm must terminate within a number of rounds equal to R). Then, by Ramsey's theorem, there exists a number $\phi(n, R)$ such that, if $|\mathcal{X}| \geq \phi(n, R)$, then there exists at least one set $\mathcal{W} \subset \mathcal{X}$ of size $|\mathcal{W}| = 2n-1$ such that all its n-subsets $\mathcal{U} \subset \mathcal{W}$, $|\mathcal{U}| = n$, share the same color. Note that $\phi(n, R)$ does not depend on the specific algorithm under consideration.

We next claim that there is a set $\bar{\mathcal{U}} \subset \mathcal{W}$ of size $|\bar{\mathcal{U}}| = n$ such that any elementary algorithm behaves like a comparison-based algorithm (i.e., one can find a comparison-based algorithm yielding identical executions) on initial conditions taken from $\bar{\mathcal{U}}$. Specifically, take $\bar{\mathcal{U}}$ as the set of the n lowest values in $\mathcal{W}$. We claim that *any* two order-equivalent sets of size $m \leq n$ in $\bar{\mathcal{U}}$ yield identical executions, thus implying that the algorithm effectively emulates a comparison-based algorithms on initial values from $\bar{\mathcal{U}}$. To prove the claim, consider two sets $\bar{\mathcal{U}}_1, \bar{\mathcal{U}}_2 \in \bar{\mathcal{U}}$ of size m. Now append to $\bar{\mathcal{U}}_1$ and $\bar{\mathcal{U}}_2$ the same $n-m$ elements of $\mathcal{W} \backslash \mathcal{U}$. The two resulting n-sets belong to $\mathcal{W}$ and elements of $\bar{\mathcal{U}}_1$ and $\bar{\mathcal{U}}_2$ appear in the same position in both n-sets: therefore they are indistinguishable and, in particular, whenever the states of two nodes u and v contain sets $\bar{\mathcal{U}}_1$ and $\bar{\mathcal{U}}_2$, respectively, such nodes will output the same value for the consensus function .

The proof is then completed by using executions of a (non-comparison-based) elementary algorithm on $\bar{\mathcal{U}}$ to "construct" a comparison-based algorithm. Specifically, we construct a comparison-based algorithm whose transitions are identical to the transitions of a given (non-comparison-based algorithm) elementary algorithm on $\bar{\mathcal{U}}$, based on the order of the elements in $\bar{\mathcal{U}}$. Since no comparison-based algorithm can solve the consensus problem with $o(n \log n)$ messages, the claim follows. □

We stress the fact that the assumptions of Proposition III.4 are satisfied whenever the set of initial conditions has infinite cardinality (e.g., $\mathcal{X} = \mathbb{R}$ or $\mathcal{X} = \mathbb{N}$).

C. Byte complexity

To prove bounds on byte complexity, we must be a little bit more specific about the content of the messages. In particular, we will assume that messages carry the UID of the sender and/or of the receiver. In practice, virtually all wireless communication protocols require each message to carry a UID identifying the sender. In addition, in non-broadcast communication protocols (as those considered in this section), messages also need to carry a receiver UID. The proof of Propositions are omitted: they follow quite easily from Propositions III.2 and III.4 (and the fact that a UID requires $\log n$ bytes to be transmitted).

Proposition III.5 (Lower bound on byte complexity for sparse networks)**.** *Assume messages carry the sender and/or the receiver UIDs. Let f be a hierarchically computable and either locally-sensitive or extractive consensus function and $\mathcal{G}$ be a set of graphs with n nodes and $|E| < n \log n$. There exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then BC$(f, \mathcal{G}) \in \Omega\Big((n \log n) \log n + n\, b\Big)$, where b is the size (in bytes) of an initial condition in $\mathcal{X}$.*

Proposition III.6 (Lower bound on byte complexity for dense networks)**.** *Assume messages carry the sender and/or the receiver UIDs. Let $\mathcal{G}$ be a set of graphs with n nodes and $|E| \geq n \log n$. Then BC$(f, \mathcal{G}) \in \Omega\Big(|E| \log n + n\, b\Big)$, where b is the size (in bytes) of an initial condition in $\mathcal{X}$.*

D. Lower bounds for local broadcast algorithms

The lower bounds on message complexity in Sections III-B and III-C are derived under the assumption of directional communication (the lower bounds on time complexity is, instead, general). This section adapts those bounds to the case of local broadcast communication. The proofs for Propositions III.7 and III.8 are omitted: they are a simple consequence of the fact that on ring topologies local broadcasts only offer a twofold improvement in message complexity. Note that, in this case, we do *not* make a distinction between dense and sparse graphs.

Proposition III.7 (Lower bound on broadcast message complexity)**.** *Let f be a locally-sensitive or extractive consensus function and $\mathcal{G}$ be a set of graphs with n nodes.There exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then bMC$(f, \mathcal{G}) \in \Omega(n \log n)$.*

[3]Informally, the trace is the history of an execution the algorithm: for each time step, it records all agents' states and all messages exchanged. For a formal definition, we refer the reader to [7]

Proposition III.8 (Lower bound on broadcast byte complexity)**.** *Assume messages carry the sender and/or the receiver UIDs. Let f be a locally-sensitive or extractive consensus function and $\mathcal{G}$ be a set of graphs with n nodes. There exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then bBC$(f, \mathcal{G}) \in \Omega(n \log^2 n + nb)$.*

IV. Tightness of Lower Bounds on Achievable Performance

In this section we study the tightness of the bounds derived in Section III.

1) Time complexity: The bound on time complexity is tight and is simply achieved by a *flooding algorithm*, which repeatedly transmits its initial value and *all* received information to its neighbors (this result is well-known in the context of leader election – its extension to our setting is rather straightforward).

Proposition IV.1 (Tightness of time complexity)**.** *For a given consensus function f and class of graphs $\mathcal{G}$ with n nodes, TC$(f, \mathcal{G}) \in \Theta(n)$.*

Proof. Omitted due to page limitations. □

2) Message complexity: Remarkably, a slight variant of the GHS algorithm [12], [13] (which builds a minimum spanning tree) achieves message optimality both for dense and for sparse graphs.

Proposition IV.2. *Let f be a locally-sensitive or extractive consensus function; let $\mathcal{G}_d$ be a set of graphs with n nodes and $|E| \geq n \log n$ and let $\mathcal{G}_s$ be a set of graphs with n nodes and $|E| < n \log n$. Then MC$(f, \mathcal{G}_d) \in \Theta(|E|)$. Furthermore, there exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then MC$(f, \mathcal{G}_s) \in \Theta(n \log n)$.*

Proof. Consider the following slight variant of the GHS algorithm. First, a rooted minimum spanning tree (MST) is constructed by executing the GHS algorithm. This operation requires $O(n \log n + |E|)$ messages. Note that at the end of the GSH algorithm the node that is the root of the MST is aware of this fact. The root node then request from all nodes their initial values. Given the tree structure, every node in a graph is contacted with at most $n-1$ messages. After a node is contacted, it forward its initial value (possibly through other nodes in the tree) to the root. Again, this operation requires at most $n - 1$ messages. Finally, the root computes the consensus function and send the consigns value to all nodes in the tree (with at most n messages). The claim then follows. □

3) Byte complexity: To prove tightness of the byte complexity bound, we need to assume that the locally-sensitive consensus function or extractive consensus function are hierarchically computable.

Proposition IV.3. *Assume messages carry the sender and/or the receiver UIDs. Let f be a locally-sensitive or extractive consensus function that is hierarchically computable, let $\mathcal{G}_d$ be a set of graphs with n nodes and $|E| \geq n \log n$ and let $\mathcal{G}_s$ be a set of graphs with n nodes and $|E| < n \log n$. Then BC$(f, \mathcal{G}_d) \in \Theta(|E| \log n + nb)$. Furthermore, there exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then BC$(f, \mathcal{G}_s) \in \Theta((n \log n) \log n + nb)$.*

Proof. Omitted due to page limitations. □

4) Tightness of bounds for local broadcast communication: The study of the tightness of the bounds for local broadcast communication hinges upon a somewhat more intricate variation of the GHS algorithm. The details are omitted due to space limitations and we only present the results.

Proposition IV.4 (Broadcast message complexity of consensus)**.** *Let f be a locally-sensitive or extractive consensus function and $\mathcal{G}$ the set of graphs with node set n. There exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then bMC$(f, \mathcal{G}) \in \Theta(n \log n)$.*

Lemma IV.5 (Broadcast byte complexity of consensus)**.** *Assume messages carry the sender and/or the receiver UIDs. Let f be a locally-sensitive or extractive consensus function and $\mathcal{G}$ a the set of graphs with n nodes. Tthere exists a function $\psi(n, R)$ such that, if the cardinality of $\mathcal{X}$ is greater than or equal to $\psi(n, R)$, then bBC$(f, \mathcal{G}) \in \Theta(n \log^2 n + nb))$.*

V. Discussion and conclusions

In Table I we provide a synoptic view of our results. Table I also includes results for D-connected networks, that is networks where the edge set is time-varying and there exists a constant $D \in \mathbb{R}_{>0}$ (possibly unknown to the nodes) such that the union of all edges appearing in the time interval $[t, t + D)$ constitute a *connected* graph. Due to page limitations, we omit the proofs for the bounds pertaining to D-connected networks. We mention, however, that such bounds can be derived with techniques very similar to the ones used in this paper.

Table I elucidates the relative advantages of different approaches to distributed computation. On static networks, the modified versions of the GHS algorithm discussed in this paper *simultaneously* achieve optimal time, message, byte, and storage complexity under mild assumptions regarding the consensus function, both for directional and broadcast communication, and both for sparse and dense graphs. On the other hand, the GHS algorithm is not readily applicable to time-varying networks (in particular, to D-connected networks) and is sensitive to single-points of failure (since it relies on the construction of a spanning tree). In other words, a GHS algorithm has minimal robustness margins to the disruption of a communication channel.

A flooding algorithm is trivially time-optimal, but as one can expect has poor message and byte complexity. Also the storage complexity is worst among all considered algorithms. On the other hand, a flooding algorithm is maximally robust to communication disruptions. Also, somewhat surprisingly, this study shows how a simple flooding algorithm outperforms an average-based algorithm (that solves the *specific* consensus problem with consensus function $f(x) = \frac{1}{n}(1, \ldots, 1)^T x$) with respect to all performance metrics except storage complexity (which, however, is arguably a minor concern for modern embedded systems).

Finally, the hybrid cluster algorithm introduced by the authors in [8] has performance intermediate between those of flooding and GHS, as a function of a tuning parameter m.

	Time	Message (S)	Message (D)	Msg. (broadcast)
Lower bound	n	$n \log n$	n^2	$n \log n$
Flooding (no failures)	**n**	$n^2 \log n$	n^3	n^2
GHS modified (no failures)	**n** [13]	$\mathbf{n \log n}$	$\mathbf{n^2}$	$\mathbf{n \log n}$
Avg. based (no failures, ε)	$n^2 \log(1/\varepsilon)$	$n^3 \log n$	n^4	n^3
Hybrid clustering[8]	$n \log(n/m)$	$2mn + k\|E_c\|m$	$2mn + k\|E_c\|m$	-
Flooding (D-connectivity)	**nD**	$n^2 D \log n$	$n^3 D$	$n^2 D$
Avg.-based (D-connectivity)	$n^2 D \log(1/\varepsilon)$	$n^3 D \log n$	$n^4 D$	$n^3 D$

	Byte (S)	Byte (D)	Byte (broadcast)	Storage
Lower bound	$(n \log n)(\log n + b)$	$n^2(\log n + b)$	$(n \log n)(\log n + b)$	$\log n + b$
Flooding (no failures)	$n^2 \log n(\log n + b)$	$n^3(\log n + b)$	$n^2(\log n + b)$	$n(\log n + b)$
GHS modified (no failures)	$(\mathbf{n \log n})(\log \mathbf{n} + \mathbf{b})$	$\mathbf{n^2}(\log \mathbf{n} + \mathbf{b})$	$(\mathbf{n \log n})(\log \mathbf{n} + \mathbf{b})$	$\log \mathbf{n} + \mathbf{b}$
Avg. based (no failures, ε)	$n^3 \log n(\log n + b)$	$n^4(\log n + b)$	$n^3(\log n + b)$	$\log n + b$
Hybrid clustering [8]	$m(n + k\|E_c\|)(\log n + b)$	$m(n + k\|E_c\|)(\log n + b)$	-	$m(\log n + b)$
Flooding (D-connectivity)	$n^3 D \log n(\log n + b)$	$n^4 D(\log n + b)$	$n^3 D(\log n + b)$	$n(\log n + b)$
Avg.-based (D-connectivity)	$n^3 D \log n(\log n + b)$	$n^4 D(\log n + b)$	$n^3 D(\log n + b)$	$\log n + b$

TABLE I: Synoptic view of bounds for distributed consensus. Bounds on time complexity hold for all consensus functions, bounds on message complexity hold for locally-sensitive or extractive consensus functions, and, finally, bounds on byte and storage complexity hold for locally-sensitive or extractive consensus functions that are hierarchically computable. S and D denote, respectively, sparse and dense graphs. For average-based algorithms, ε is the convergence threshold for termination. We denote in bold face optimality results.

The main advantage of this algorithm is to "trade" some of the optimality of GHS with a tunable "degree of robustness", as a function of m. We remark that the parameter $|E_c|$ in the table is $|E_c| = O(\min(E, m^2))$.

As discussed, GHS-like algorithms (and also the hybrid algorithm in [8]) can not be readily applied to dynamic settings. Note, however, that the execution time of GHS-like algorithms is $O(n)$, an order of magnitude lower than average-based algorithms [6]: if reconfigurations or disruptions are infrequent (i.e., their frequency is much lower than $O(1/n)$), GHS-like algorithms can indeed be applied to nominally dynamic networks.

We conclude this paper with a discussion of the limitations of our analysis, which immediately reflect into a number of interesting directions for future research. First, optimal algorithms for D-connected networks are not currently known (of course, it might be the case that our lower bounds are not tight), which represents a key area of study. Second, while locally-sensitive and extractive consensus functions represent a fairly large class of consensus functions, it is of interest to generalize our bounds on message, byte, and storage complexity even further. Third, our approach is a worst-case approach over the classes of *sparse* and *dense* graphs. An interesting direction for future research would be (i) to derive bounds on a finer partition of the class of possible graphs, for example by parameterizing the graphs by their maximum in- or out-node degree, and (ii) by embedding the problem within a probabilistic structure, to derive performance bounds with respect to sets of graphs randomly drawn from a given probability distribution, so that one can derive measures of average (as opposed to worst-case) performance. Fourth, our model essentially does not include the notion or robustness with respect to either stopping (i.e., for malfunctions) or byzantine (i.e., malicious) failures, which is an aspect of pivotal importance for the reliable deployment of cyber-physical systems. Finally, a practical implementations of the algorithms discussed in this paper could shed additional light on the relative benefits of the different approaches.

Overall, we hope that this work will prompt researchers in the field of multi-agent systems to compare their results against the fundamental lower bounds derived in this paper to properly evaluate the relative benefits of their approach.

REFERENCES

[1] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. IEEE Conf. on Decision and Control*, New Orleans, LA, Dec. 2007, pp. 5492–5498.

[2] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control: Collective group behavior through local interaction," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[3] M. de Weerdt and B. Clement, "Introduction to planning in multiagent systems," *Multiagent and Grid Systems*, vol. 5, no. 4, pp. 345–355, 2009.

[4] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[5] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[6] A. Olshevsky, "Efficient information aggregation strategies for distributed control and signal processing," Ph.D. dissertation, MIT - EECS Department, September 2010.

[7] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.

[8] F. Rossi and M. Pavone, "Decentralized decision-making on robotic networks with hybrid performance metrics," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*. IEEE, 2013, pp. 358–365.

[9] N. Cutland, *Computability: An introduction to recursive function theory*. Cambridge University Press, 1980.

[10] G. N. Frederickson and N. A. Lynch, "Electing a leader in a synchronous ring," *Journal of the Association for Computing Machinery*, vol. 34, no. 1, pp. 98–115, 1987.

[11] M. Snir, "On parallel searching," *SIAM Journal on Computing*, vol. 14, no. 3, pp. 688–708, 1985.

[12] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 5, no. 1, pp. 66–77, 1983.

[13] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 1987, pp. 230–240.

[14] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520 – 1533, Sept. 2004.

[15] E. Korach, S. Moran, and S. Zaks, "Tight lower and upper bounds for some distributed algorithms for a complete network of processors," in *Proceedings of the third annual ACM symposium on Principles of distributed computing*. ACM, 1984, pp. 199–207.

[16] F. P. Ramsey, "On a problem of formal logic," *Proceedings of the London Mathematical Society*, vol. s2-30, no. 1, pp. 264–286, 1930. [Online]. Available: http://plms.oxfordjournals.org/content/s2-30/1/264.short